NSF Award DUE-0443044 Annual Report, March 2009

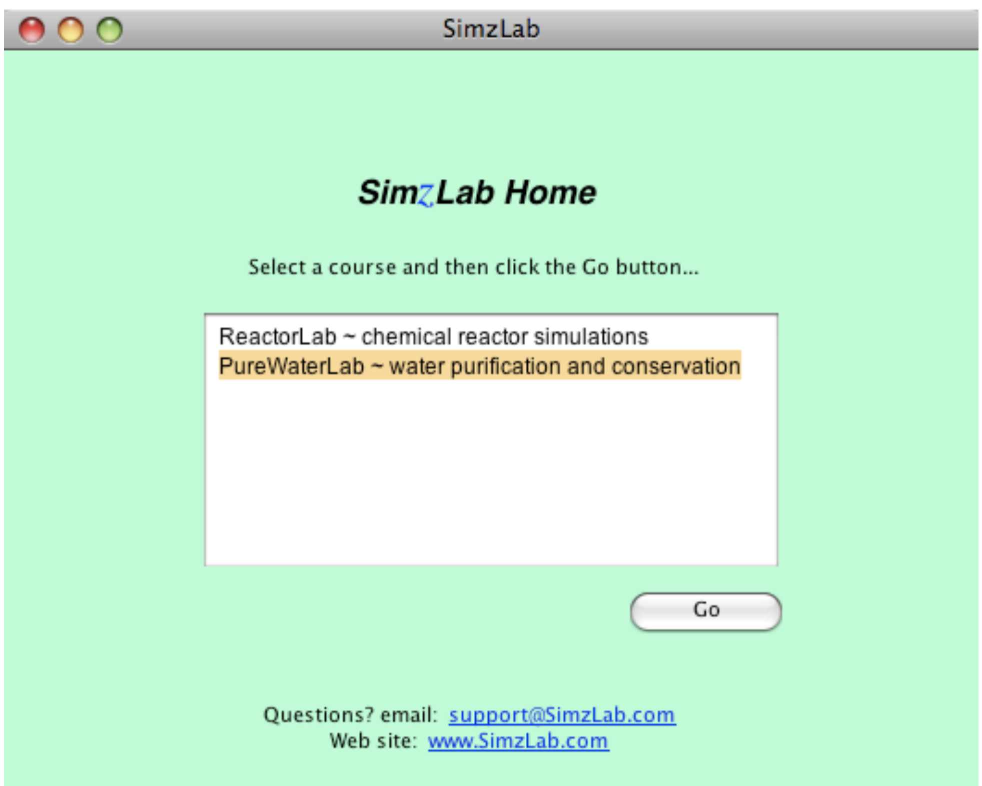
PureWaterLab - Conservation Education and Research Through Interactive Simulation

This attachment to the annual report discusses progress and plans for the PureWaterLab project.

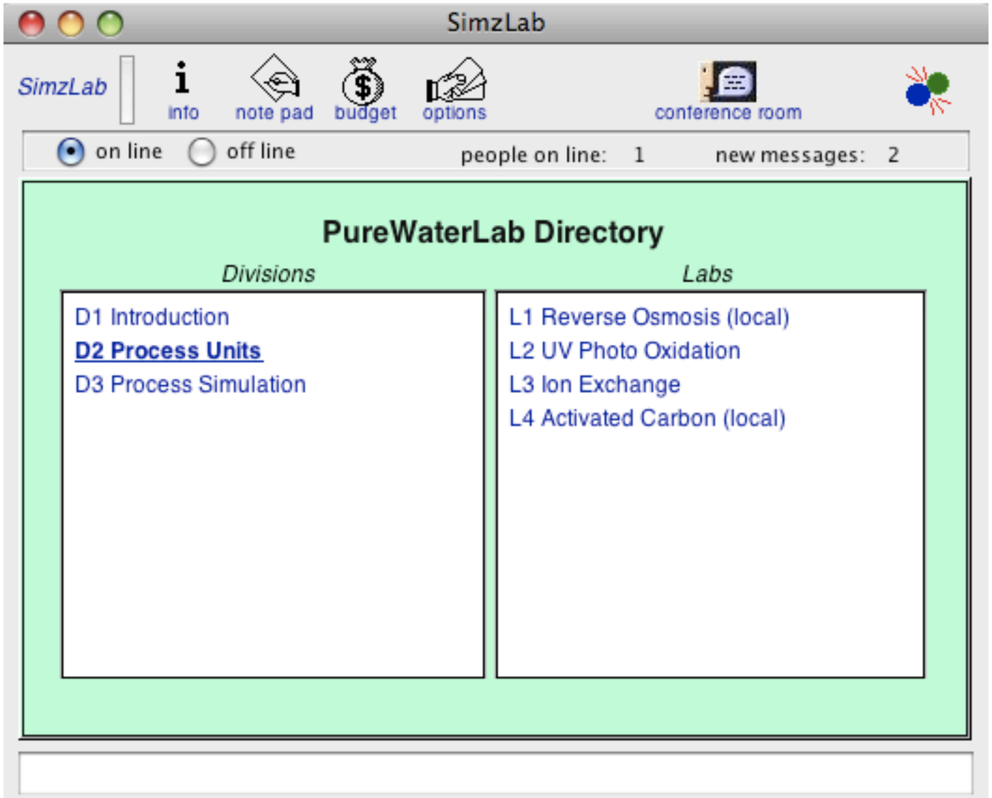
During the past year we continued work on the project in a no-cost extension. Major accomplishments include merging our software codes from two NSF-funded projects into a common code-base, developing additional course modules including those which provide detailed and compute-intensive simulations of physical systems, and continued development of our plant simulator for team collaboration.

In this project and a prior NSF-CCLI project, we developed two sets of interactive simulation modules: Reactor Lab and PureWaterLab. The software codes for these two projects were similar and developing them separately and keeping them both current was a headache. Therefore, this past year we developed a common code-base and a common software application for distribution of both sets of modules. We call this common delivery vehicle, SimzLab. The software can be downloaded at no cost from SimzLab.com.

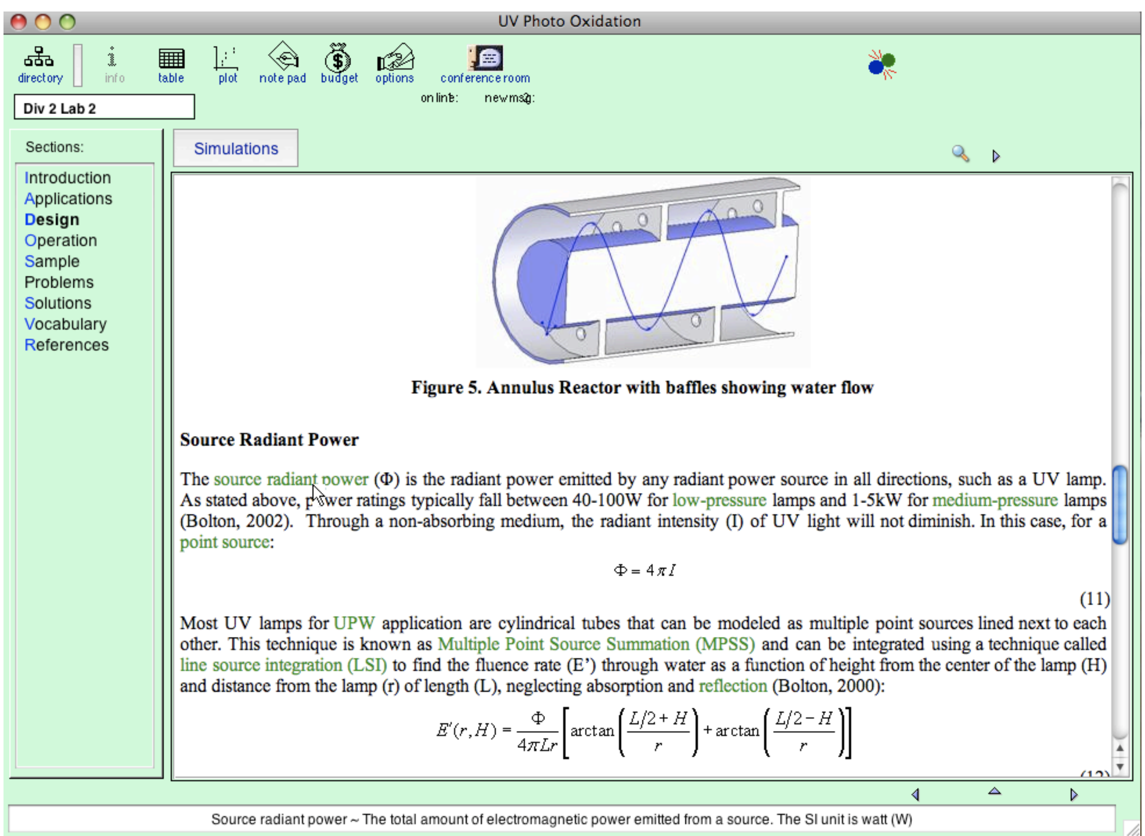
SimzLab is a desktop application that is integrated with the Internet and associated software on web servers. In the current web jargon, the Lab is a "rich Internet application." When on line, a student can access new modules and communicate in the Conference Room with other students. Updates to software are automatically downloaded and installed. When off line, the student can continue to work on the modules they previously accessed while on line. This is the home screen showing the "courses" currently available. We plan to add new courses in the future.



This is the Directory screen of PureWaterLab. The modules or "Labs" are arranged in "Divisions."



This screen shows the explanatory part of a module on the UV Photo Oxidation method of water purification (screens are not shown at same scale in this document).

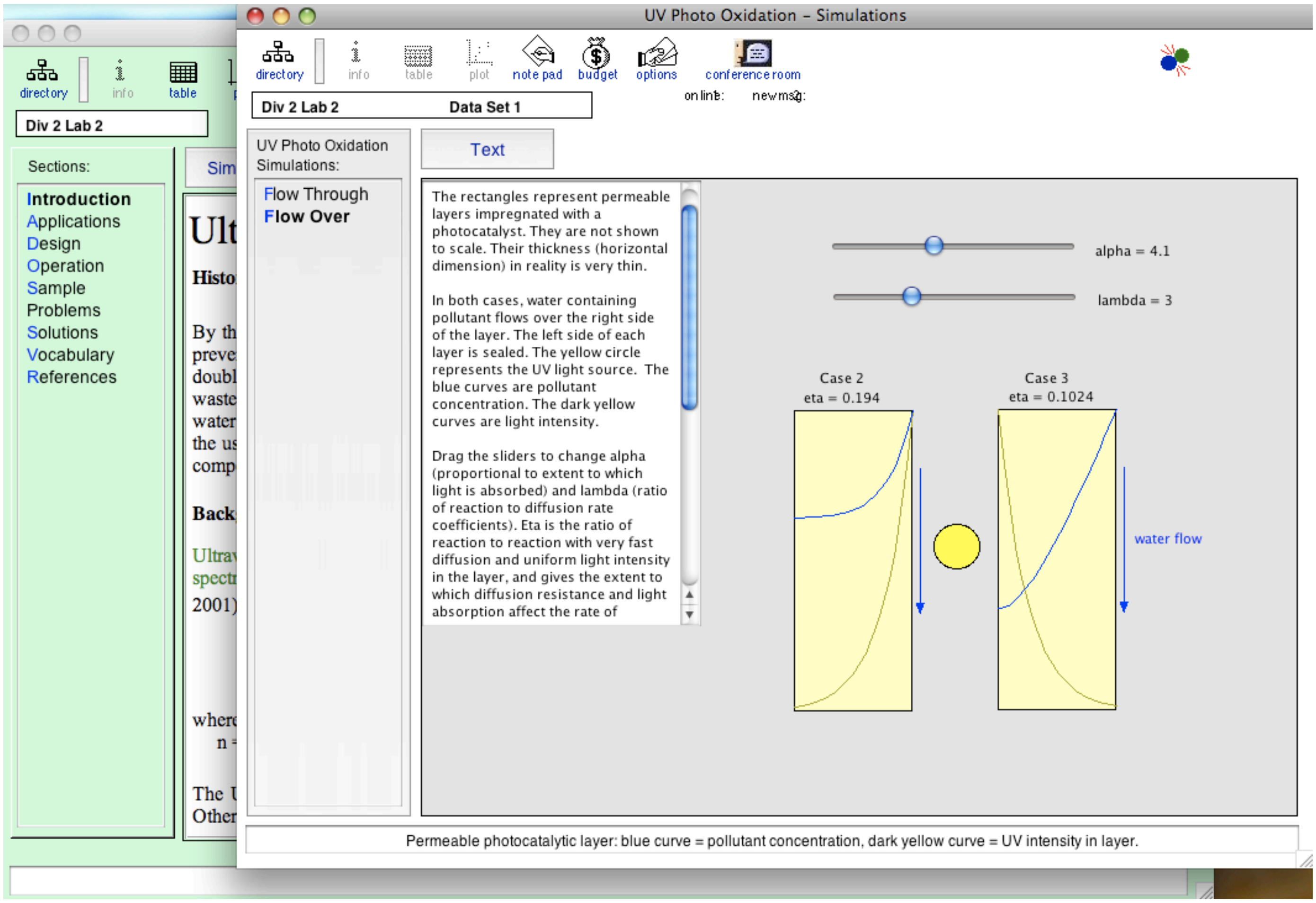


This project is a collaboration. The University of California, San Diego (UCSD) part of the team is working on the software programming and the interactive simulations. The University of Arizona (UA) part of the team is working on the main module content, including text, graphics, math equations, and assessment components.

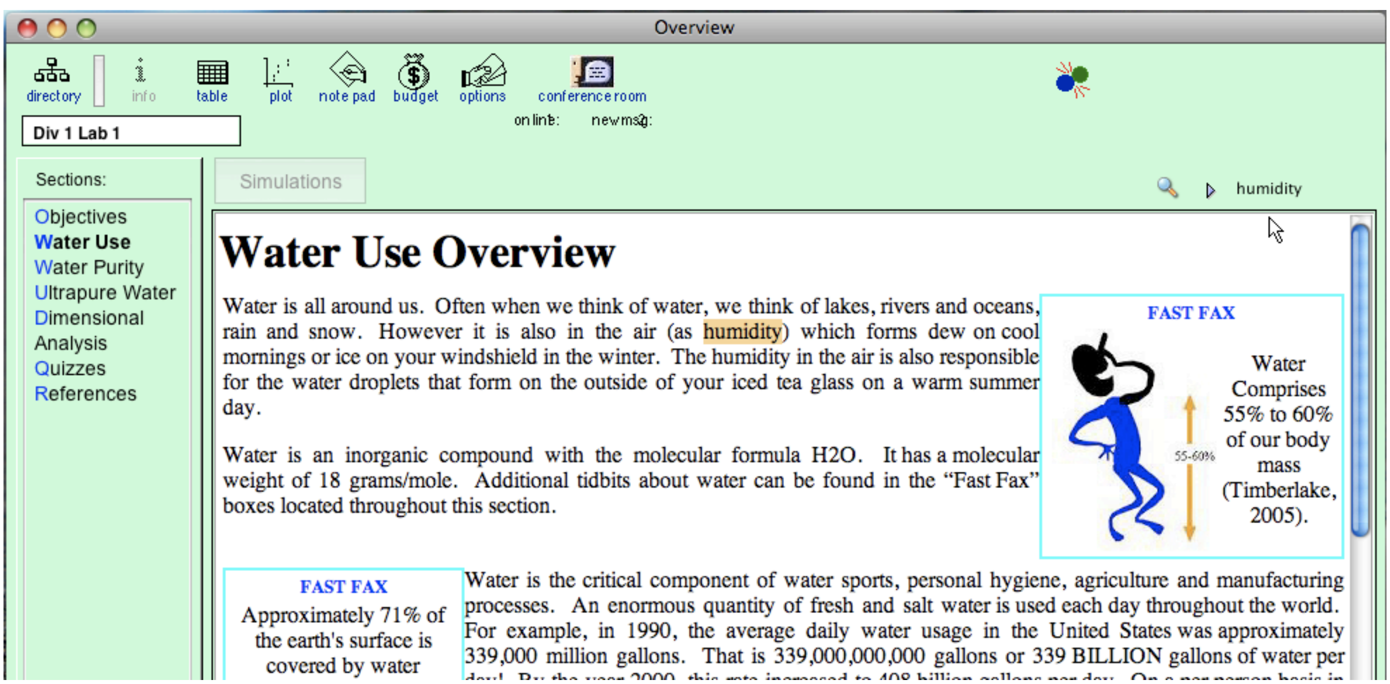
A work process was developed such that the UA group can develop and add new and revised content easily without having to involve the software group at UCSD. The UA group develops content as standard web pages and uploads them to the PureWaterLab server. Whenever a student is using PureWaterLab on-line, the software automatically detects new and updated modules and downloads them for on- or off-line use.

Text can contain links to web sites external to PureWaterLab. These links are opened in the users web browser.

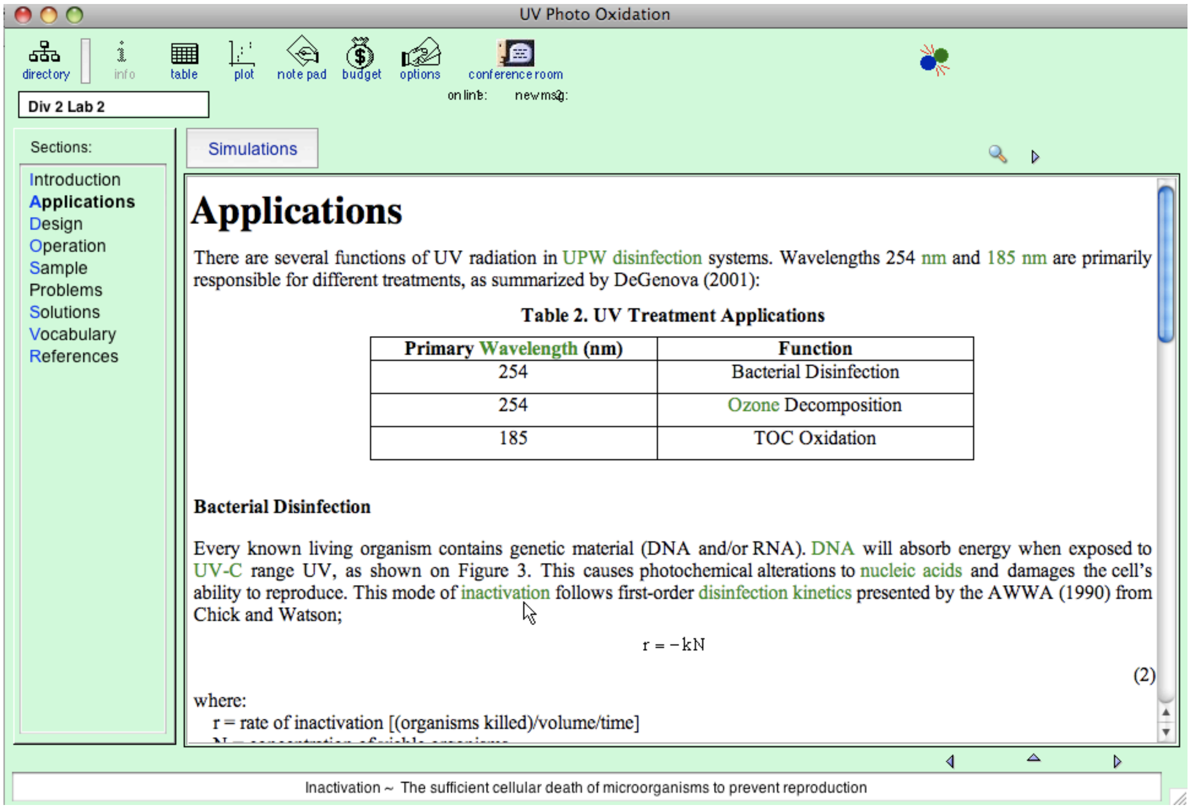
The advantage of using these web pages in PureWaterLab, as compared with a standard web browser, is that many other features are additional available in PureWaterLab, such as the interactive simulations. A simulation in the UV Photo Oxidation module is shown below.



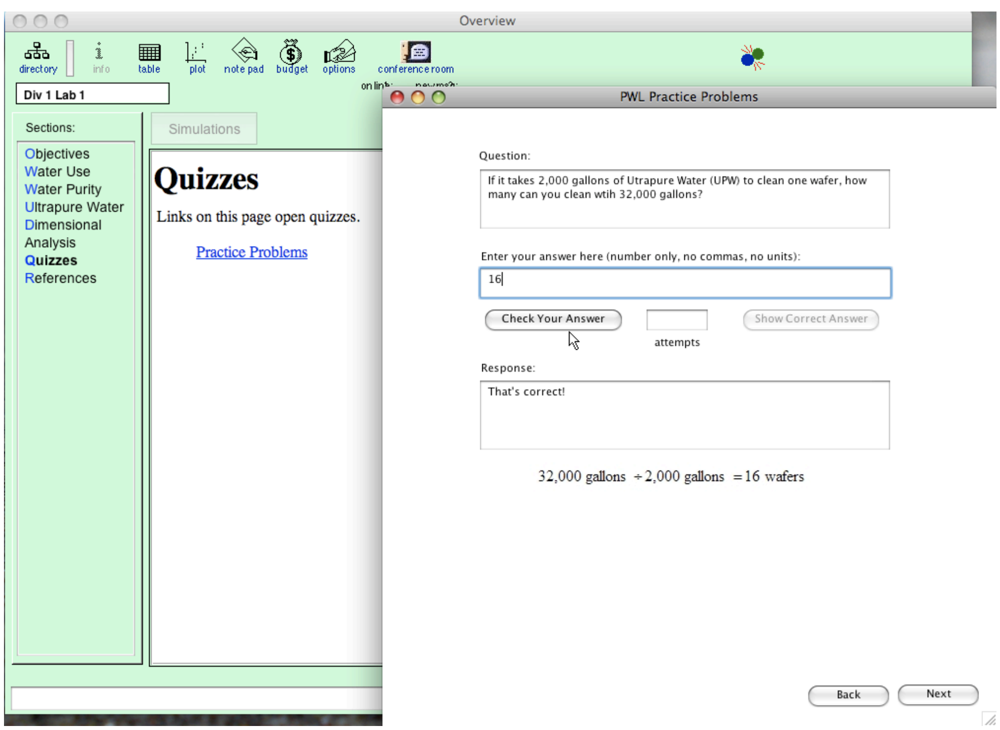
Several ways were developed to help students search text and understand vocabulary. A result of a search for a word is shown here.



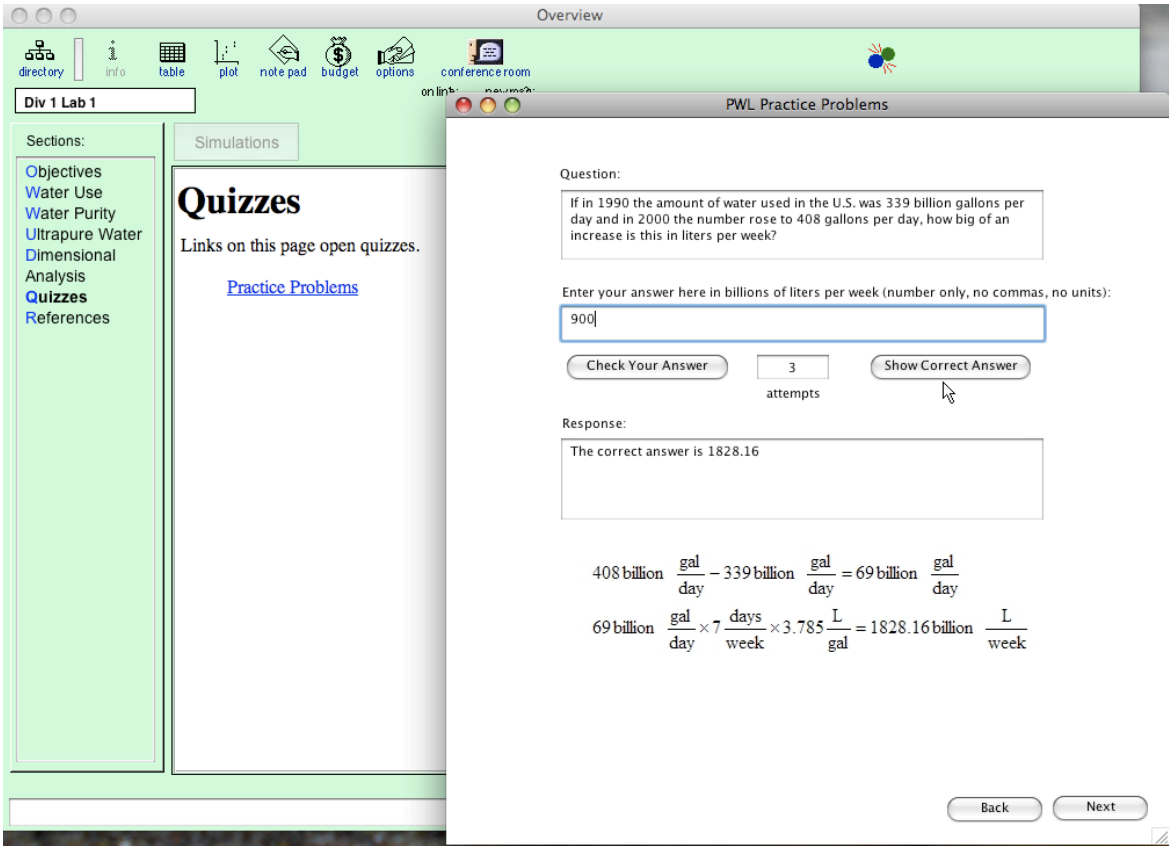
One special feature provided by the software is automatic scanning of text for words listed in the vocabulary section. Other than preparing the vocabulary section, the content authors do not have to do anything else. The software highlights vocabulary words automatically, and the definition is shown at the bottom of the window when the student passes the cursor over a highlighted word, as shown below.



Another new feature being added are quizzes to assess student learning. Several different types of questions are available, and the software automatically scores the answers and provides feedback, as shown here.



If a student enters incorrect answers several times, they are allowed to view the correct answer.



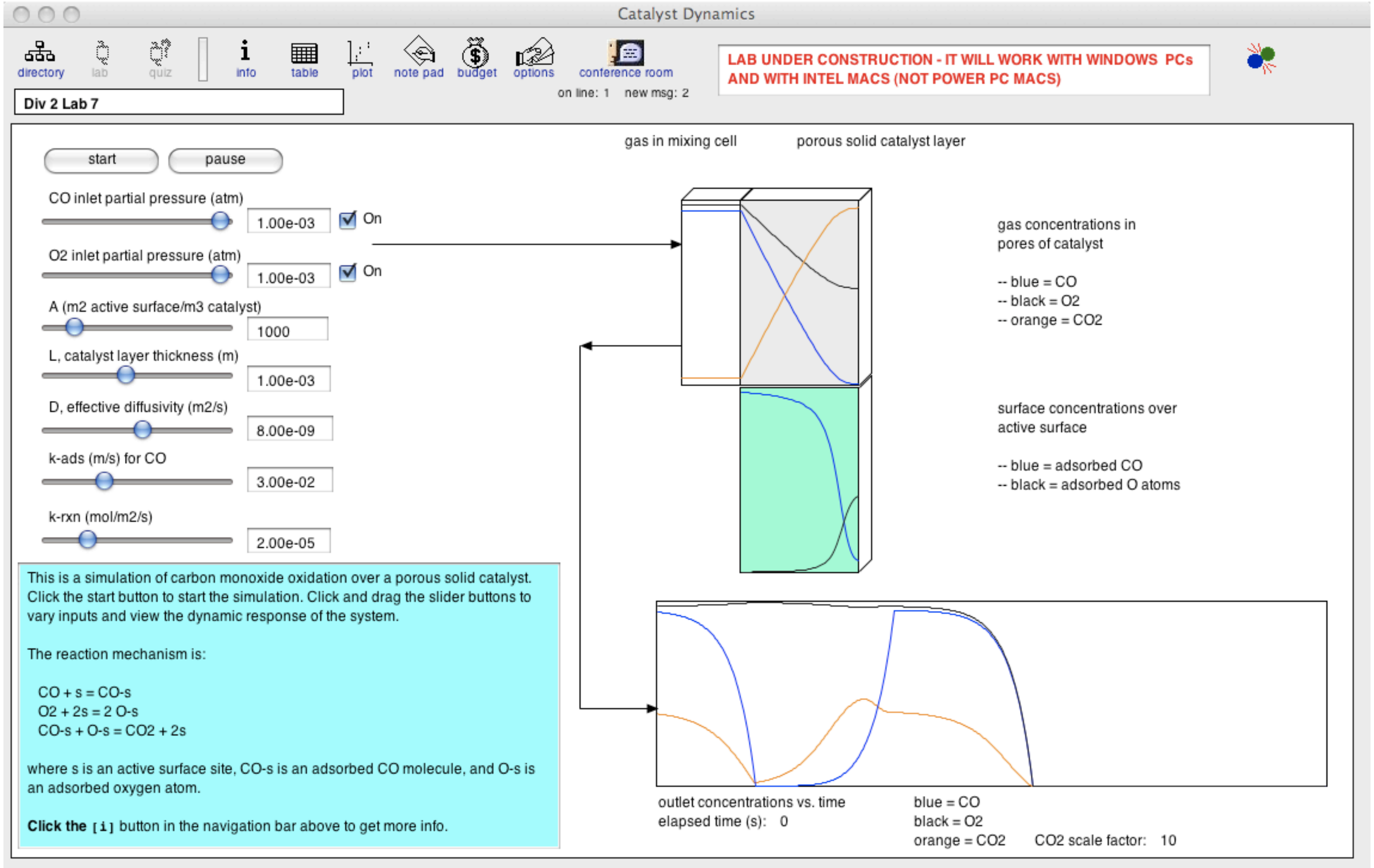
One of the developments in 2008-2009 was development of interactive simulations which can provide very detailed, compute-intensive simulations of physical processes. Whereas the cross-platform software language that is used for most of the project has sufficient speed to handle most simulations, compiled executables need to be used for more demanding simulations.

One of our visions is to provide undergraduate students with research-grade simulations where appropriate. Our development this year in this area was to couple simulations in C++ compiled executable files to the Graphical User Interface or GUI and also to deliver those compiled files along with the other module components.

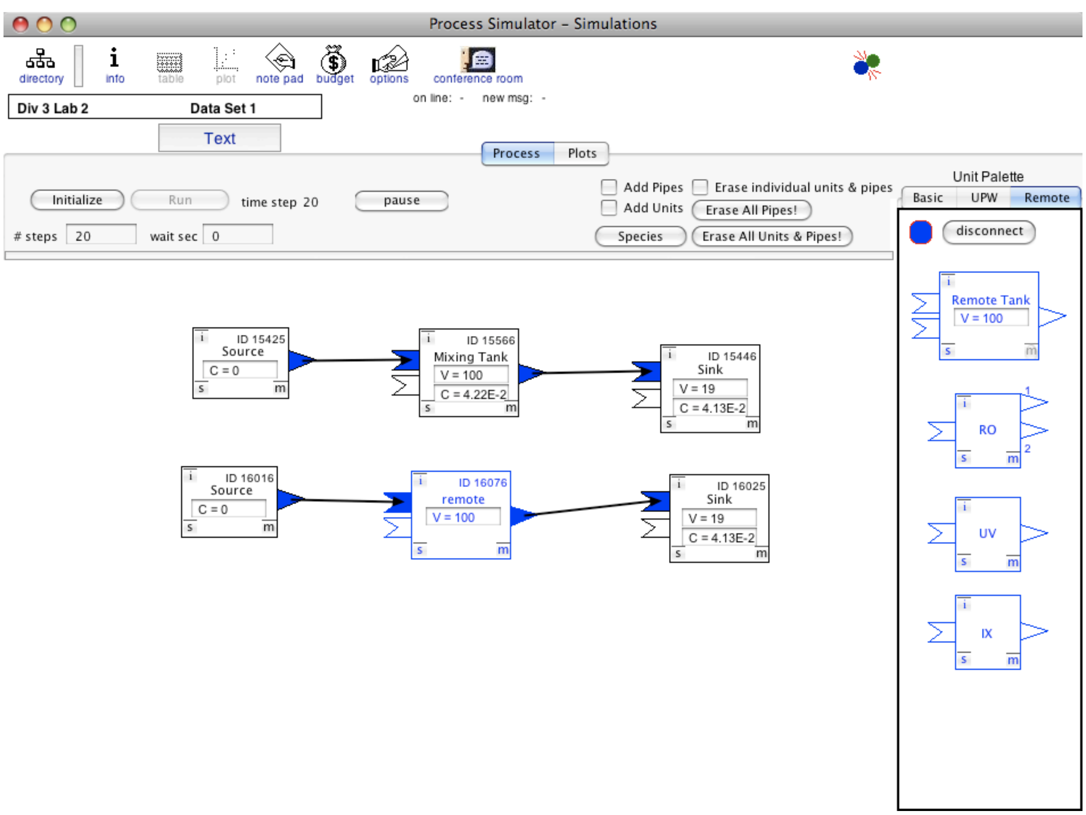
The figure on the next page shows a dynamic simulation in Reactor Lab of the CO oxidation reaction in a porous solid catalyst. When running, the concentration profiles in the gas mixing cell, the pores and surface of the catalyst layer, and the outlet gas concentrations continuously update. The student can change parameters and the gas inputs by moving the sliders.

At each time step in the GUI, the module sends the current values of the inputs to the compiled executable file which then integrates the system equations for many internal time steps, returns the output to the GUI, and then the GUI updates the plots.

This procedure will be used to develop other detailed simulations.



Progress is continuing to be made on the plant simulator to allow for inter-campus collaboration on simulations. The figure below shows the simulator with two processes in parallel.



Each process consists of a water source, a pipe to a mixing tank, and a pipe to a sink. There is a contaminant component in the water leaving the source, whose concentration "C" is shown.

In the top process, all units are "local" in the sense that they are computed on the client computer on which this copy of PureWaterLab is resident. In the bottom process, the blue unit is a local proxy for the actual unit which resides on a remote computer. When running, the water flowing through the pipes is carried by "messages" which are passed from unit (software object) to unit. Messages sent to a local proxy for a remote unit are sent over the Internet to the remote unit via TCP/IP socket messages. The remote unit updates its state and sends a return message back to its proxy on the original computer.

Messages sent between local units are written in the language of the local client. Messages sent across the Internet are written in cross-platform XML text. Because these messages are written in XML, the different simulators on the different computers which participate in a simulation can be written in any computer language and run on any operating system on any hardware.

The major progress on the simulator during the last year was to speed up the execution of the simulation and to test running collaborative simulations between UCSD and U. Arizona.

Development of the simulator will continue. Our vision is to enable student groups at different universities to collaborate on designing and running plants. During a collaborative simulation, students will be able to communicate via text messages using the technology developed for the Lab's current Conference Room.

Some grant funds remain and we remain committed to development of the project, so we are requesting a no-cost extension for an additional year. Continuation will allow the graduate student at UCSD to finish the PhD degree and will allow undergraduate students at U. Arizona to develop more course modules and test the modules in courses.